GroupReport

Akshayaswuroupikka Balasubramanian, Yixuan Xu

February 11, 2016

Assignment 1: Optimizing a model parameter

The file mortality_rate.csv contains information about mortaily rates of the fruit flies during a certain period.

1.1

The data is successsfully imported in R.One more variable LMR is added to the data which is the natural logarithm of Rate.After which the data is divided into training and test sets by using the following code:

```
n=dim(data)[1]
set.seed(123456)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]
```

1.2

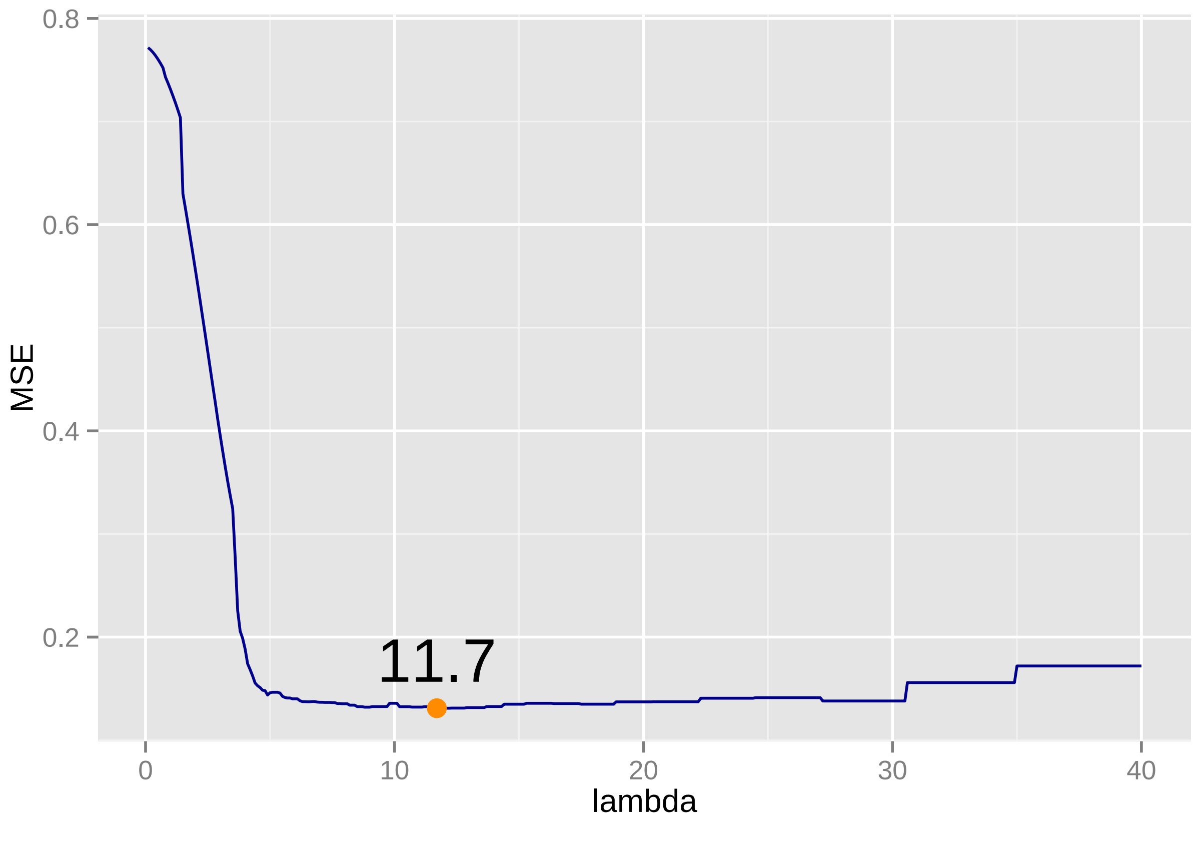
The function myMSE() is written for given parameters λ and list pars containing vectors X, Y, Xtest, Ytest.This fits a LOESS model with response Y and predictor X using loess() function with penalty lambda (parameter enp.target in loess()) and then predicts the model for Xtest. The function computes the predictive MSE,prints it and also returns as a result.

```
myMSE <- function(lambda,pars){
  model=loess(pars$Y~pars$X,enp.target = lambda)
  Predictor=predict(model,newdata=pars$Xtest)
  MSE=(1/length(pars$Ytest))*sum((Predictor-pars$Ytest)^2)
  print(MSE)
  return(MSE)
}
```

1.3

The function myMSE() is used to estimate the predictive MSE values with training and test sets and with the following lambda values: . Lambda=0.1,0.2,.,40 A plot of the predictive MSE values against the respective lambda is shown below:

```
## [1] 0.7716498
## [1] 0.7696322
## [1] 0.7670135
## [1] 0.763903
## [1] 0.7603617
## [1] 0.7564285
## [1] 0.7521297
## [1] 0.7430915
## [1] 0.7373437
## [1] 0.7312544
## [1] 0.724835
## [1] 0.7180934
## [1] 0.7110349
## [1] 0.703662
## [1] 0.6299142
## [1] 0.6158768
## [1] 0.6014798
## [1] 0.5867457
## [1] 0.5716976
## [1] 0.5563601
## [1] 0.5407604
## [1] 0.5249291
## [1] 0.5089018
## [1] 0.4927201
## [1] 0.4764333
## [1] 0.4600991
## [1] 0.4437853
## [1] 0.4275698
## [1] 0.4110532
## [1] 0.3953247
## [1] 0.3800023
## [1] 0.3651851
## [1] 0.3509584
## [1] 0.3373794
## [1] 0.324461
## [1] 0.2772786
## [1] 0.2255412
## [1] 0.2057829
## [1] 0.1987935
## [1] 0.1883268
## [1] 0.1741243
## [1] 0.1685396
## [1] 0.162383
## [1] 0.1554774
## [1] 0.1528338
## [1] 0.1510906
## [1] 0.1484618
## [1] 0.1481252
## [1] 0.1438308
## [1] 0.1461178
## [1] 0.1464905
## [1] 0.1464905
## [1] 0.1465163
## [1] 0.145769
## [1] 0.142542
## [1] 0.1414151
## [1] 0.1409552
## [1] 0.1409552
## [1] 0.1401702
## [1] 0.1402161
## [1] 0.1402161
## [1] 0.1383069
## [1] 0.1374353
## [1] 0.1374353
## [1] 0.1373248
## [1] 0.1373248
## [1] 0.1375201
## [1] 0.1375201
## [1] 0.1369977
## [1] 0.1367946
## [1] 0.1367946
## [1] 0.1366528
## [1] 0.1366528
## [1] 0.1366528
## [1] 0.1365055
## [1] 0.1365055
## [1] 0.1355826
## [1] 0.1355826
## [1] 0.1354196
## [1] 0.1354196
## [1] 0.1354196
## [1] 0.1340062
## [1] 0.1340062
## [1] 0.1340062
## [1] 0.1326046
## [1] 0.1326046
## [1] 0.1326046
## [1] 0.1320531
## [1] 0.1320531
## [1] 0.1320531
## [1] 0.1326215
## [1] 0.1326215
## [1] 0.1326215
## [1] 0.1326215
## [1] 0.1326581
## [1] 0.1326581
## [1] 0.1326581
## [1] 0.135802
## [1] 0.135802
## [1] 0.135802
## [1] 0.135802
## [1] 0.1325542
## [1] 0.1325542
## [1] 0.1325542
## [1] 0.1325542
## [1] 0.1325542
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1325391
## [1] 0.1325391
## [1] 0.1325391
## [1] 0.1325391
## [1] 0.1325391
## [1] 0.131047
## [1] 0.131047
## [1] 0.131047
## [1] 0.131047
## [1] 0.131047
## [1] 0.131047
## [1] 0.1311756
## [1] 0.1311756
## [1] 0.1311756
## [1] 0.1311756
## [1] 0.1311756
## [1] 0.1311756
## [1] 0.1316359
## [1] 0.1316359
## [1] 0.1316359
## [1] 0.1316359
## [1] 0.1316359
## [1] 0.1316359
## [1] 0.1316359
## [1] 0.1316359
## [1] 0.1326683
## [1] 0.1326683
## [1] 0.1326683
## [1] 0.1326683
## [1] 0.1326683
## [1] 0.1326683
## [1] 0.1326683
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1349211
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1358018
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1355502
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1349612
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1372244
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1373502
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1407325
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1412809
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1380524
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1558879
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
```



The optimal lambda value is obtained when lambda is equal to 11.7.As we are calculating the predictive MSE values from 0.1 to 40 by steps of 0.1, 400 evaluations of myMSE were required to find the optimal value for lambda.

1.4

The optimize() function is used in which the range for search and accuracy is specified as [0.1,40],0.01.

```
## [1] 0.1358018
## [1] 0.1412809
## [1] 0.1326581
## [1] 0.1401702
## [1] 0.1325391
## [1] 0.1321441
## [1] 0.1325542
## [1] 0.1321441
## [1] 0.1325391
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
## [1] 0.1321441
```

```
## $minimum
## [1] 10.69361
##
## $objective
## [1] 0.1321441
```

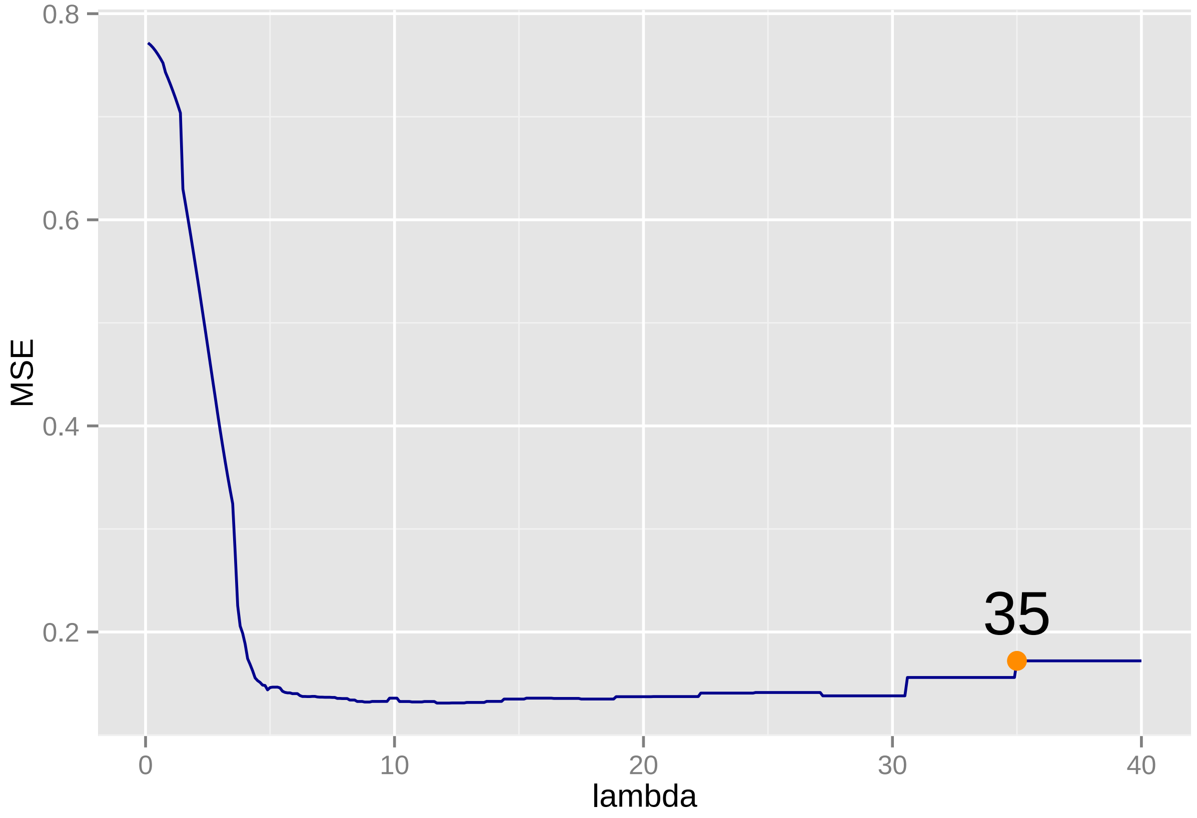
The optimal MSE was found by optimize().Compared to the step 3, the value is a little bit larger, but it could be considered as the same when you keep the two significand number. The evaluations have a big difference,by counting the number of values printed it is concluded that 18 evaluations were needed.

1.5

The optim() function and BFGS method with starting point lambda=35 is used to find the optimal lambda value.

```
## [1] 0.1719996
## [1] 0.1719996
## [1] 0.1719996
```

```
## $par
## [1] 35
##
## $value
## [1] 0.1719996
##
## $counts
## function gradient
##        1        1
##
## $convergence
## [1] 0
##
## $message
## NULL
```



The result gives that 3 evaluations of myMSE were required to find the optimal lambda value.In this case, the optimal MSE is a local optimum.BFGS method is particularly good for large-scale problems, but there is a problem when MSE drop in local optimum the algorithm will believe that it is global optimum(i.e. it will find the MSE values close to that point and comes to a result that it is the global optimum and it will not look for a better solution).

Assignment 2: Maximizing likelihood

File data.RData contains a sample from normal distribution with some parameters μ, σ.

2.1

The data is loaded in to R environment.

2.2

Maximum likelihood:

$$f(X_1, X_2...X_{100} \mid \mu, \sigma) =$$

$$\prod \frac{1}{\sigma\sqrt{2\pi}} exp \frac{-(x_i - \mu)^2}{(2\sigma^2)} =$$

$$\frac{(2\pi)^{-\frac{n}{2}}}{\sigma^n} exp(-\frac{\sum(x_i - \mu)^2}{2\sigma^2})$$

log-likelihood:

$$log(f) = -\frac{n}{2}\,log(2\pi) - n\,log(\sigma) - \frac{\sum(x_i-\mu)^2}{2\sigma^2}$$

Differentiate with respect to σ, μ:

$$\frac{\partial(log(f))}{\partial\mu} = \frac{\sum_1^n(x_i-\mu)}{\sigma^2} = 0$$

$$\frac{\partial(log(f))}{\partial\sigma} = -\frac{n}{\sigma} + \frac{\sum(x_i-\mu)^2}{\sigma^3} = 0$$

The partials need to be equal to zero, and solve:

$$\hat{\mu} = \frac{\sum x_i}{n}$$

$$\hat{\sigma} = \sqrt{\frac{\sum(x_i-\hat{\mu})^2}{n}}$$

```
mu
```

```
## [1] 1.275528
```

```
sigma
```

```
## [1] 2.005976
```

2.3

It is often mathematically more tractable to maximise a sum of functions rather than a product of function. Therefore, instead of trying to maximise the likelihood function people prefer to maximise the log-likelihood function.

2.4

The minus log-likelihood function is optimized with initial parameters $\mu = 0, \sigma = 1$.Both Conjugate Gradient method and BFGS algorithm with gradient specified and without is executed below:

Conjugate Gradient method:

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##      210       37
##
## $convergence
## [1] 0
##
## $message
## NULL
```

BFGS algorithm:

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       37       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Conjugate Gradient method with gradient specified:

```
## $par
## [1] 1.275528 2.005976
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       56       17
##
## $convergence
## [1] 0
##
## $message
## NULL
```

BFGS algorithm with gradient specified:

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       41       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

2.5

All cases are convergence to 0.

Conjugate Gradient method: $\sigma = 1.275528$, $\mu = 2.005977$, function evaluations are 210, gradient evaluation are 37.

BFGS algorithm: $\sigma = 1.275528$, $\mu = 2.005977$, function evaluations are 37, gradient evaluation are 15.

Conjugate Gradient method with gradient specified: $\sigma = 1.275528$, $\mu = 2.005976$, function evaluations are 56, gradient evaluation are 17.

BFGS algorithm with gradient specified: $\sigma = 1.275528$, $\mu = 2.005977$, function evaluations are 41, gradient evaluation are 15.

All these method could get the same σ and μ (only Conjugate Gradient method with gradient specified, μ is a little bit different, but we would consider that it is the same).So we would choose the BFGS algorithm without gradient specified as it has the lowest evaluation in both function and gradient.

Appendix - R-code

```
## ----eval=FALSE----------------------------------------------------------
## n=dim(data)[1]
## set.seed(123456)
## id=sample(1:n, floor(n*0.5))
## train=data[id,]
## test=data[-id,]

## ----echo=FALSE----------------------------------------------------------
#import the data to R
mortality<-read.csv2("mortality_rate.csv",sep=";")
#adding a new column
mortality$LMR<-log(mortality$Rate)
#dividing the data
n=dim(mortality)[1]
set.seed(123456)
id=sample(1:n, floor(n*0.5))
train=mortality[id,]
test=mortality[-id,]

## --------------------------------------------------------------------
myMSE <- function(lambda,pars){
  model=loess(pars$Y~pars$X,enp.target = lambda)
  Predictor=predict(model,newdata=pars$Xtest)
  MSE=(1/length(pars$Ytest))*sum((Predictor-pars$Ytest)^2)
  print(MSE)
  return(MSE)
}

## ----echo=FALSE----------------------------------------------------------
this_mortality=list(X=train$Day, Y=train$LMR, Xtest=test$Day, Ytest=test$LMR)
lambda=seq(from=0.1,to=40,by=0.1)
this_mse=0
for(i in 1:length(lambda)){
  this_mse[i]=myMSE(lambda[i],this_mortality)
}
library(ggplot2)
MSEdata <- data.frame(MSE=this_mse, lambda=lambda)
ggplot(MSEdata, aes(x=lambda, y=MSE, label=lambda)) + geom_line(col="darkblue")+
  geom_text(data=subset(MSEdata, MSE <= min(MSEdata$MSE))[1,],vjust=-0.6, size=8) +
  geom_point(data=subset(MSEdata, MSE <= min(MSEdata$MSE))[1,], size=3.5, col="darkorange")

## ----echo=FALSE-----------------------------------------------------------
Optim=optimize(f=myMSE,interval=c(0.1, 40),tol=0.01,pars=this_mortality)
Optim

## ----echo=FALSE-----------------------------------------------------------
optim(par=35, fn=myMSE, method = "BFGS", pars=this_mortality)
ggplot(MSEdata, aes(x=lambda, y=MSE, label=lambda)) + geom_line(col="darkblue")+
  geom_text(data=subset(MSEdata,  MSEdata$lambda == 35),vjust=-0.6, size=8) +
  geom_point(data=subset(MSEdata,  MSEdata$lambda == 35), size=3.5, col="darkorange")

## ---- echo=FALSE----------------------------------------------------------
load("data.RData")


## ---- echo=FALSE----------------------------------------------------------
mu <- mean(data)
sigma <- sqrt(mean((data-mu)^2))


## ---------------------------------------------------------------------------
mu
sigma

## ---- echo=FALSE----------------------------------------------------------
minus_loglikelihood <- function(parameters, data){
  n <- length(data)
  mu <- parameters[1]
  sigma <- parameters[2]
  minus_ll <- -(-(n/2)*log(2 * pi) - (n/2)*log(sigma^2) - sum((data - mu)^2/(2*sigma^2)))
  return(minus_ll)
}

gradient_specified <- function(parameters, data){
  n <- length(data)
  mu <- parameters[1]
  sigma <- parameters[2]
  def_mu <- -sum((data - mu)/sigma^2)
  def_sigma <- n/sigma-sum((data - mu)^2)/(sigma^3)
  return(c(def_mu,def_sigma))
}

## ---- echo=FALSE----------------------------------------------------------
optim(c(0,1) , minus_loglikelihood, method = "CG", data=data)

## ---- echo=FALSE----------------------------------------------------------
```

```
optim(c(0,1) , minus_loglikelihood, method = "BFGS", data=data)

## ---- echo=FALSE---------------------------------------------------------
optim(c(0,1) , fn = minus_loglikelihood, gr = gradient_specified, method = "CG", data=data)

## ---- echo=FALSE---------------------------------------------------------
optim(c(0,1) , fn = minus_loglikelihood, gr = gradient_specified, method = "BFGS", data=data)

## ----code=readLines(knitr::purl("GroupReport.Rmd", documentation = 1)), eval = FALSE----
## NA
```